

## CAPÍTULO III: DESARROLLO

### 3.1 Capturas de la Aplicación (Documentación completa del desarrollo, Scripts, Sprites, Prefabs e imágenes)

#### Pantalla de Inicio

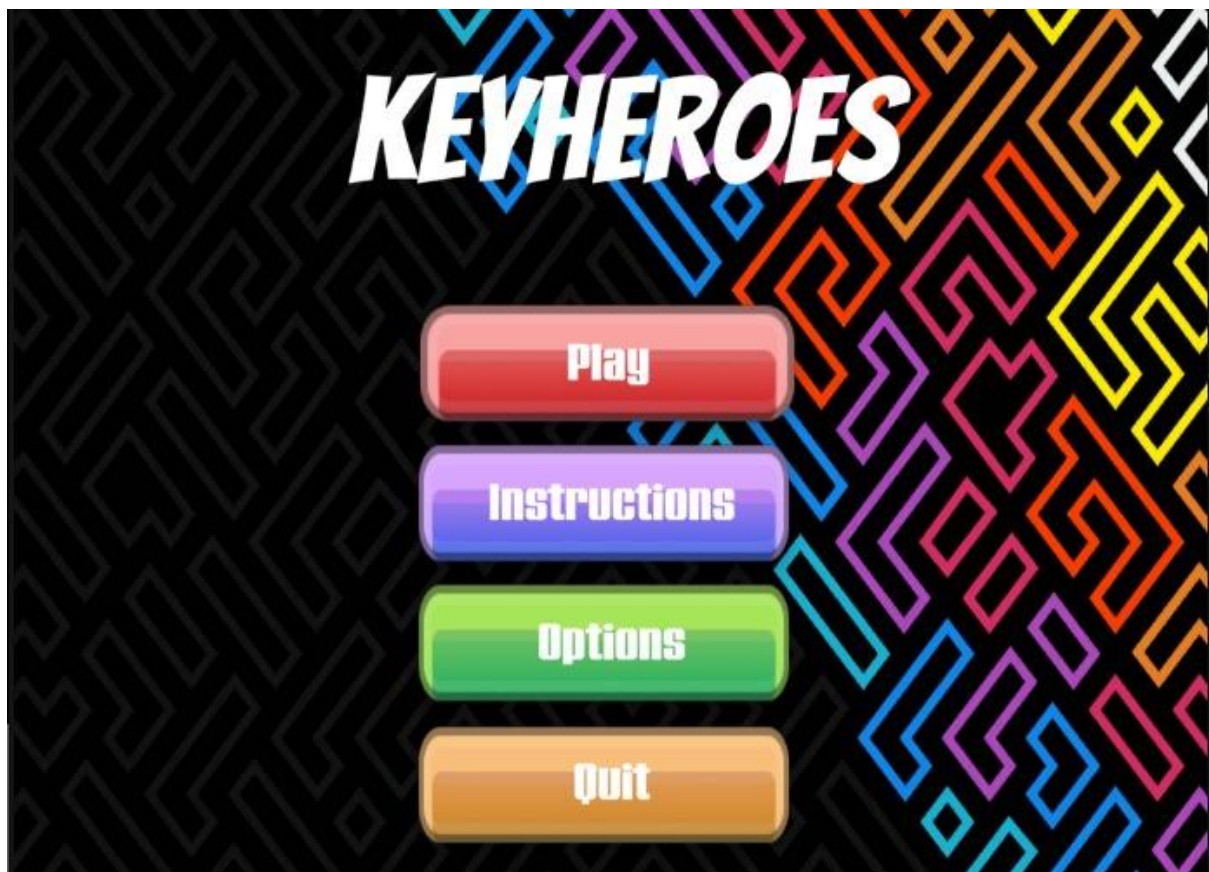
En esta pantalla se encuentran los botones de

**PLAY**( Para ir al menu de eleccion)

**INSTRUCTIONS**(Para mostrar las instrucciones)

**OPTIONS**(Muestra opciones y ajustes)

**QUIT**(Cierra el juego)



## INSTRUCCIONES:

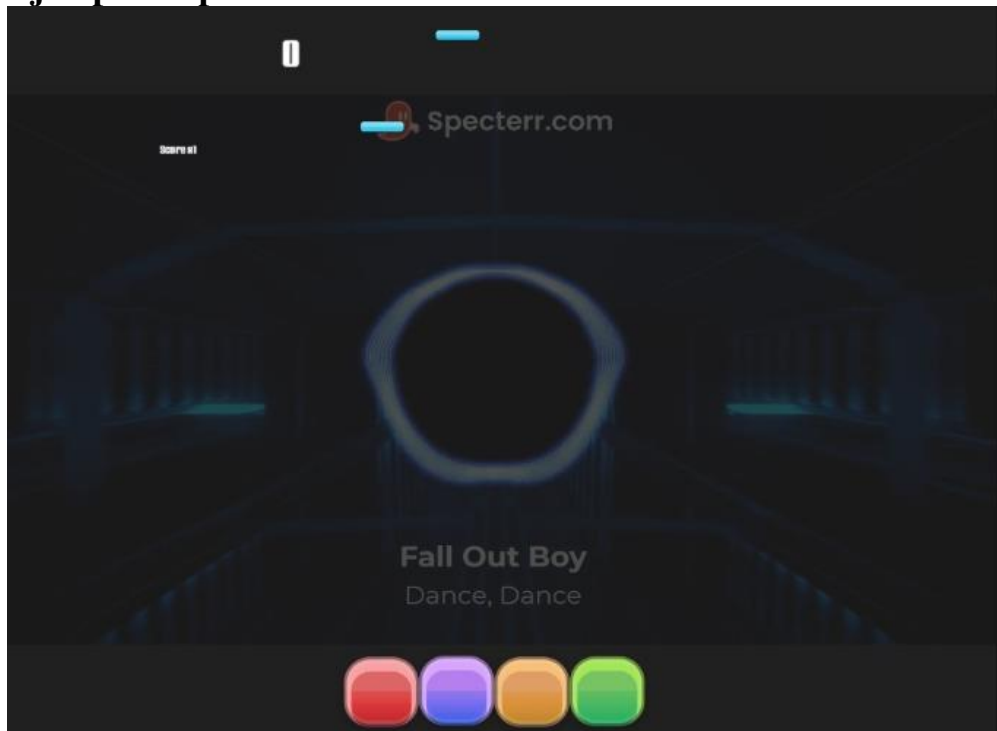


## Lista de canciones:

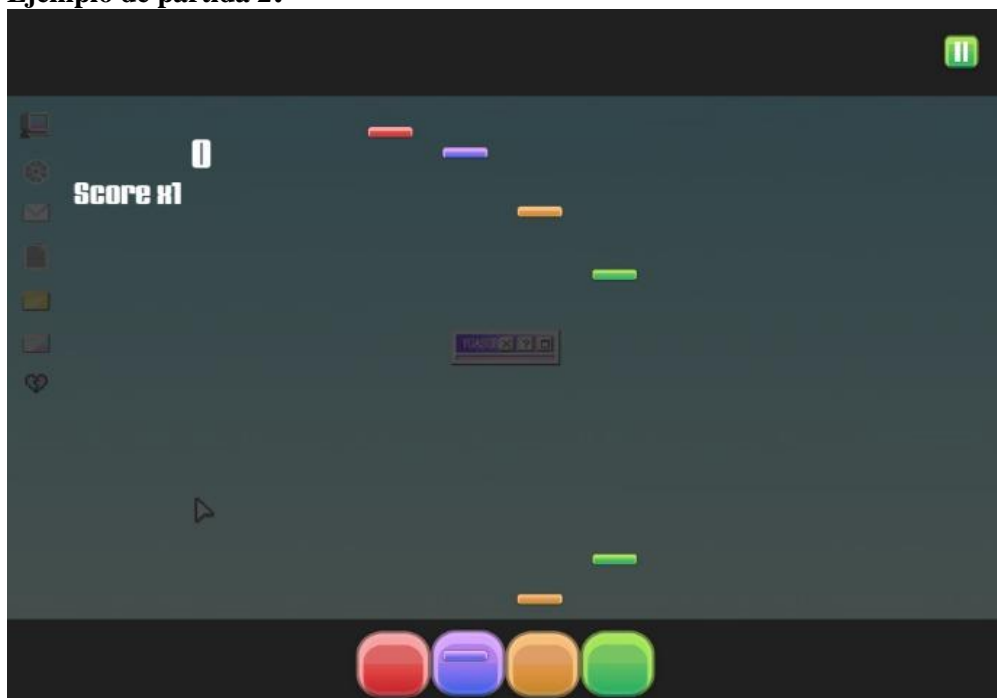
Muestra la lista de canciones disponibles para jugar



### Ejemplo de partida 1:



### Ejemplo de partida 2:



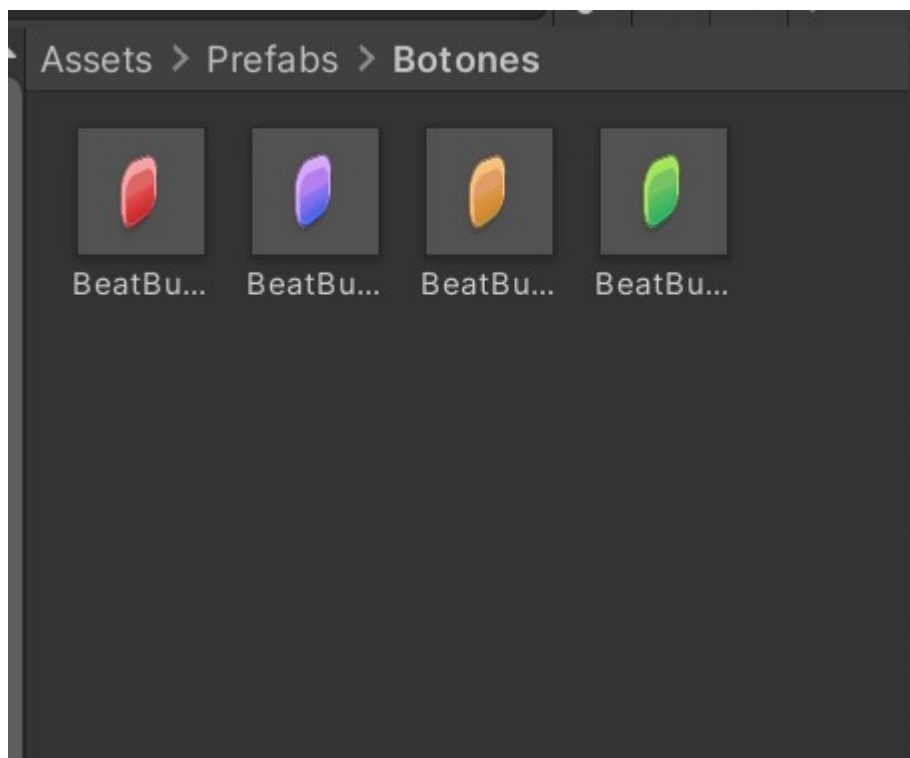
## Sprites:

Sprites utilizados para el juego, botones y notas...



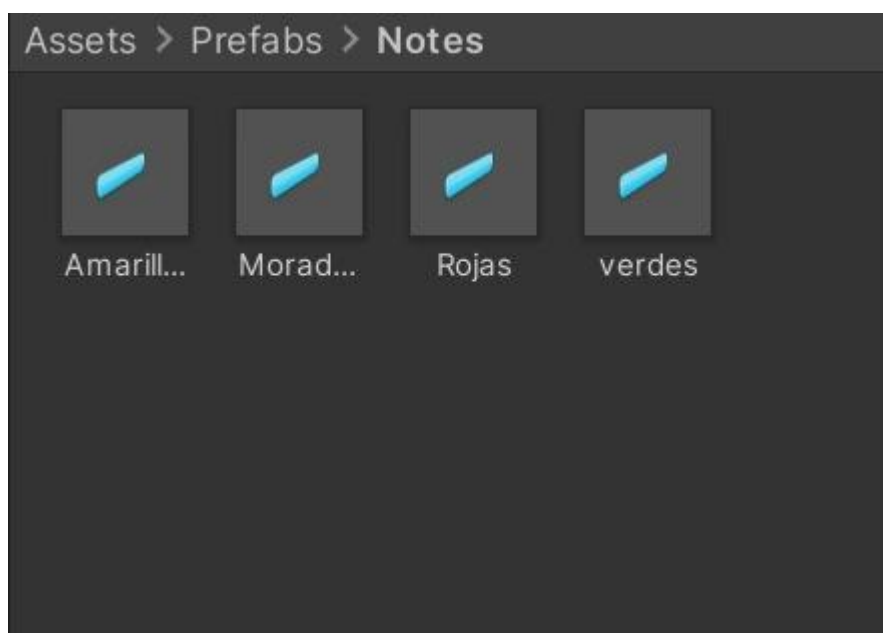
## Prefabs:

Prefabs de botones



## Prefabs:

Prefabs de notas



## SCRIPTS:

**\*Nota:** solo se incluyeron en este documento los scripts mas importantes para el juego, scripts para navegación entre pantallas y control de botones fueron omitidos para no alargar innecesariamente el documento.

Script para la lista de notas que descienden:

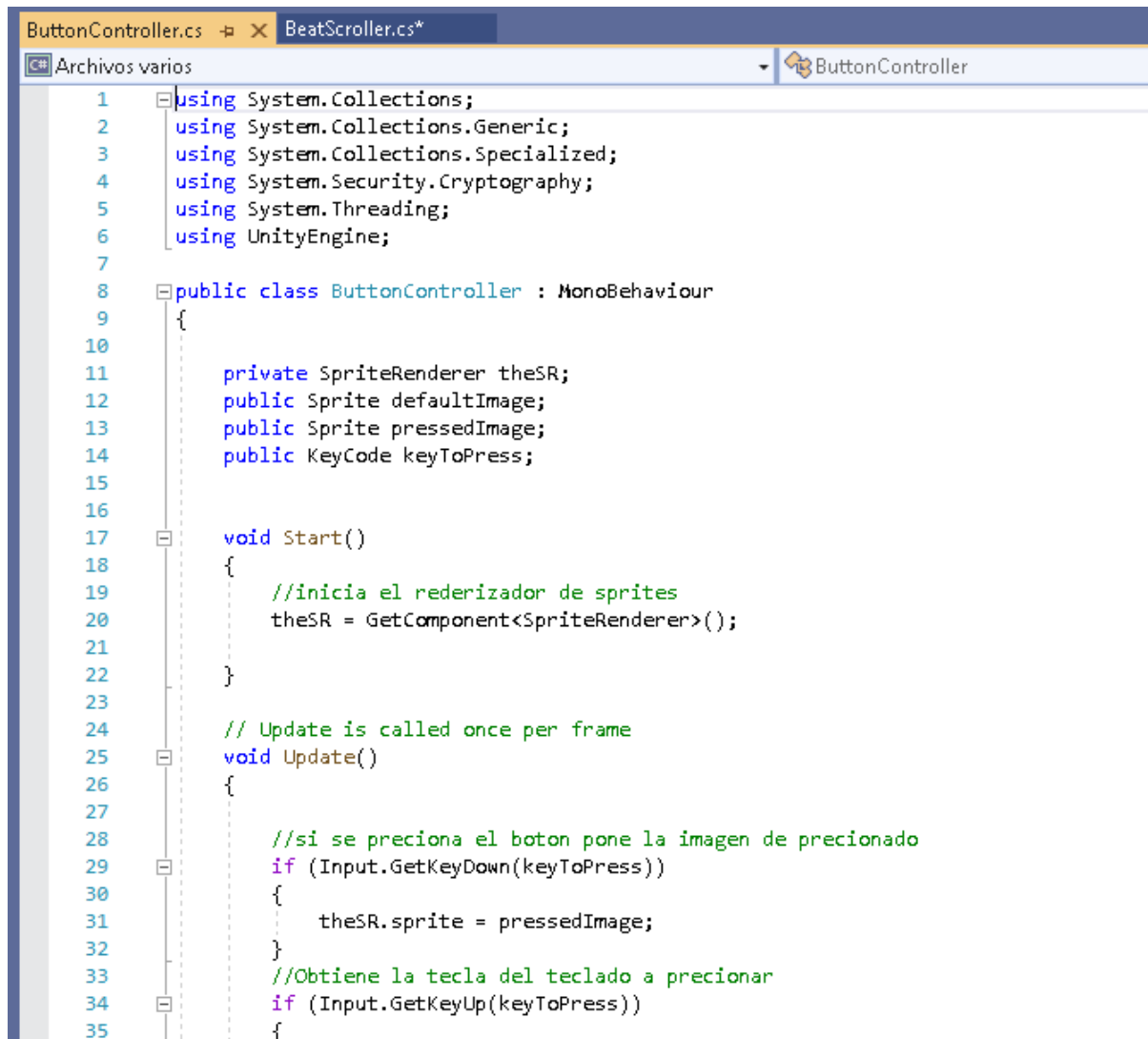
```
BeatScroller.cs*
Archivos varios BeatScroller

1  using System.Collections;
2  using System.Collections.Generic;
3  using System.Collections.Specialized;
4  using System.Security.Cryptography;
5  using System.Threading;
6  using UnityEngine;
7
8  public class BeatScroller : MonoBehaviour
9  {
10
11      public bool createMode;
12      public float beatTempo;
13      public bool hasStarted;
14      public GameObject n;
15      public KeyCode keyToPress;
16
17
18      // Start is called before the first frame update
19      void Start()
20      {
21
22
23
24          //Para el modo creador
25          //Esto pone en reversa el movimiento de la lista
26          if(createMode)
27          {
28              beatTempo = beatTempo / -60f;
29          }
30          else{
31              beatTempo = beatTempo / 60f;
32          }
33
34      }
35
```

```
BeatScroller.cs*
Archivos varios BeatScroller

34  }
35
36      // Update is called once per frame
37      void Update()
38      {
39
40          if (hasStarted)
41          {
42              if (Input.anyKeyDown)
43              {
44                  hasStarted = true;
45              }
46          }
47          else
48          {
49              //Para el modo creador
50              //Esto pone una nota cada vez que se presiona un boton
51              if(createMode)
52              {
53
54                  if(Input.GetKeyDown(keyToPress))
55                  {
56                      Instantiate(n,transform.position,Quaternion.identity);
57                  }
58                  transform.position -= new Vector3(0f, beatTempo * Time.deltaTime, 0f);
59              }
60              //Si el modo creador esta activado
61              //Esto es para el movimiento del mapa de notas
62              else
63              {
64                  transform.position -= new Vector3(0f, beatTempo * Time.deltaTime, 0f);
65              }
66          }
67      }
68
```

Script para los botones del juego:

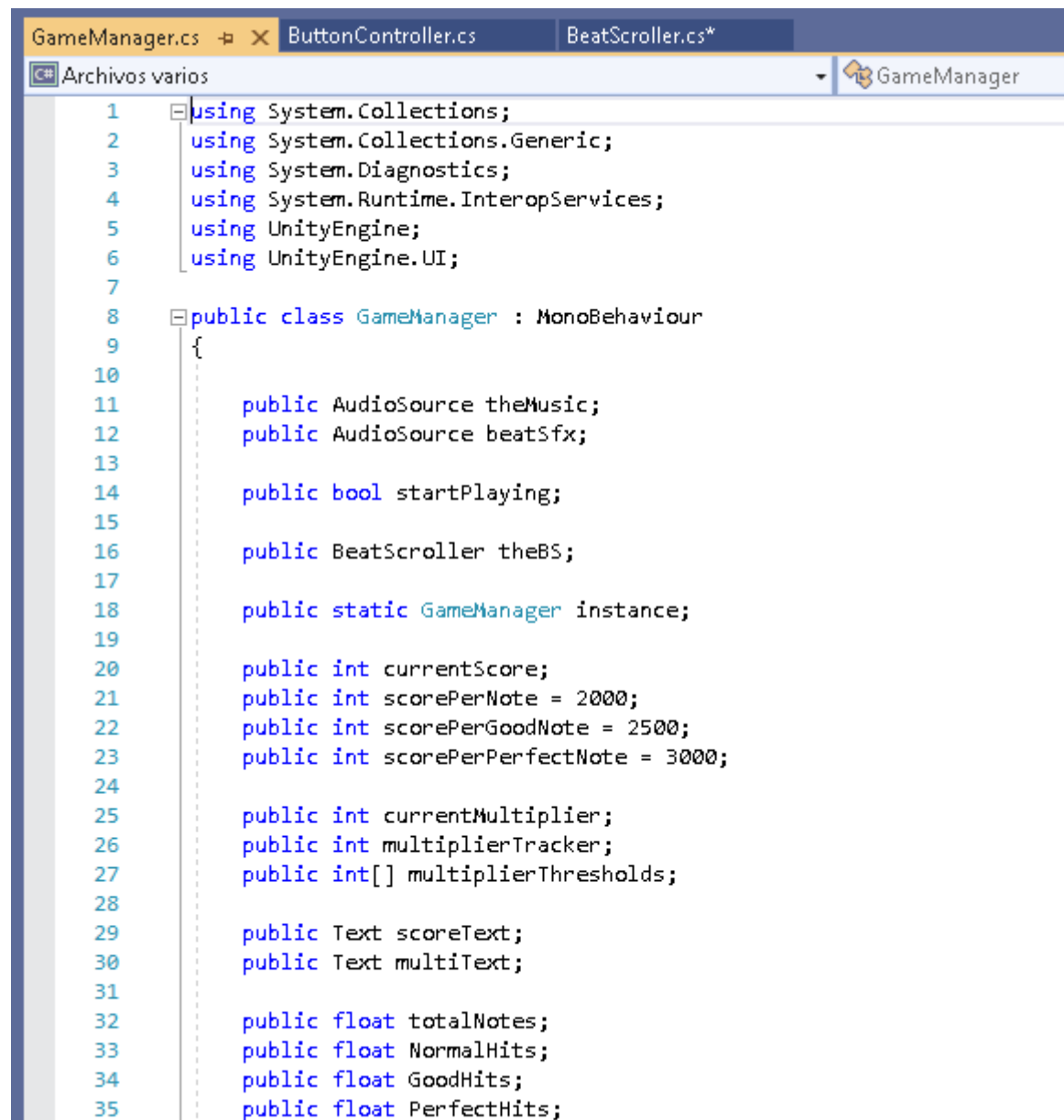


The image shows a screenshot of a Unity IDE with two tabs open: 'ButtonController.cs' and 'BeatScroller.cs\*'. The 'ButtonController.cs' tab is active, showing a C# script. The script defines a 'ButtonController' class that inherits from 'MonoBehaviour'. It includes several using statements at the top, followed by a class definition with private and public fields, and two methods: 'Start()' and 'Update()'. The 'Update()' method contains logic to change the sprite of a 'SpriteRenderer' component when a specific key is pressed or released.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using System.Collections.Specialized;
4 using System.Security.Cryptography;
5 using System.Threading;
6 using UnityEngine;
7
8 public class ButtonController : MonoBehaviour
9 {
10
11     private SpriteRenderer theSR;
12     public Sprite defaultImage;
13     public Sprite pressedImage;
14     public KeyCode keyToPress;
15
16
17     void Start()
18     {
19         //inicia el rederezador de sprites
20         theSR = GetComponent<SpriteRenderer>();
21     }
22
23     // Update is called once per frame
24     void Update()
25     {
26
27         //si se preciona el boton pone la imagen de precionado
28         if (Input.GetKeyDown(keyToPress))
29         {
30             theSR.sprite = pressedImage;
31         }
32         //Obtiene la tecla del teclado a precionar
33         if (Input.GetKeyUp(keyToPress))
34         {
35
```



Script para El controlador del juego:



```
1  using System.Collections;
2  using System.Collections.Generic;
3  using System.Diagnostics;
4  using System.Runtime.InteropServices;
5  using UnityEngine;
6  using UnityEngine.UI;
7
8  public class GameManager : MonoBehaviour
9  {
10
11     public AudioSource theMusic;
12     public AudioSource beatSfx;
13
14     public bool startPlaying;
15
16     public BeatScroller theBS;
17
18     public static GameManager instance;
19
20     public int currentScore;
21     public int scorePerNote = 2000;
22     public int scorePerGoodNote = 2500;
23     public int scorePerPerfectNote = 3000;
24
25     public int currentMultiplier;
26     public int multiplierTracker;
27     public int[] multiplierThresholds;
28
29     public Text scoreText;
30     public Text multiText;
31
32     public float totalNotes;
33     public float NormalHits;
34     public float GoodHits;
35     public float PerfectHits;
```



```
GameManager.cs  ButtonController.cs  BeatScroller.cs*
Archivos varios  GameManager

31
32 public float totalNotes;
33 public float NormalHits;
34 public float GoodHits;
35 public float PerfectHits;
36 public float MissedHits;
37
38 public GameObject resultsScreen;
39 public Text percentHitText, normalsText, goodsText, perfectsText, missesText, rankText, finalScoreText;
40
41
42
43 // Start is called before the first frame update
44 // Para iniciar la partida
45 void Start()
46 {
47     instance = this;
48     scoreText.text = "Preciona Una Tecla Para Empezar";
49
50     currentMultiplier = 1;
51     totalNotes = ((FindObjectsOfType<NoteObject>()).Length) - 168;
52 }
53
54 // Update is called once per frame
55 void Update()
56 {
57     if (!startPlaying)
58     {
59         //Detecta cuando se preciona una tecla para iniciar
60         if (Input.anyKeyDown)
61         {
62             scoreText.text = "0";
63             startPlaying = true;
64             theBS.hasStarted = true;
65             theMusic.Play();
66         }
67     }
68 }
```

```
GameManager.cs  ButtonController.cs  BeatScroller.cs*
Archivos varios  GameManager

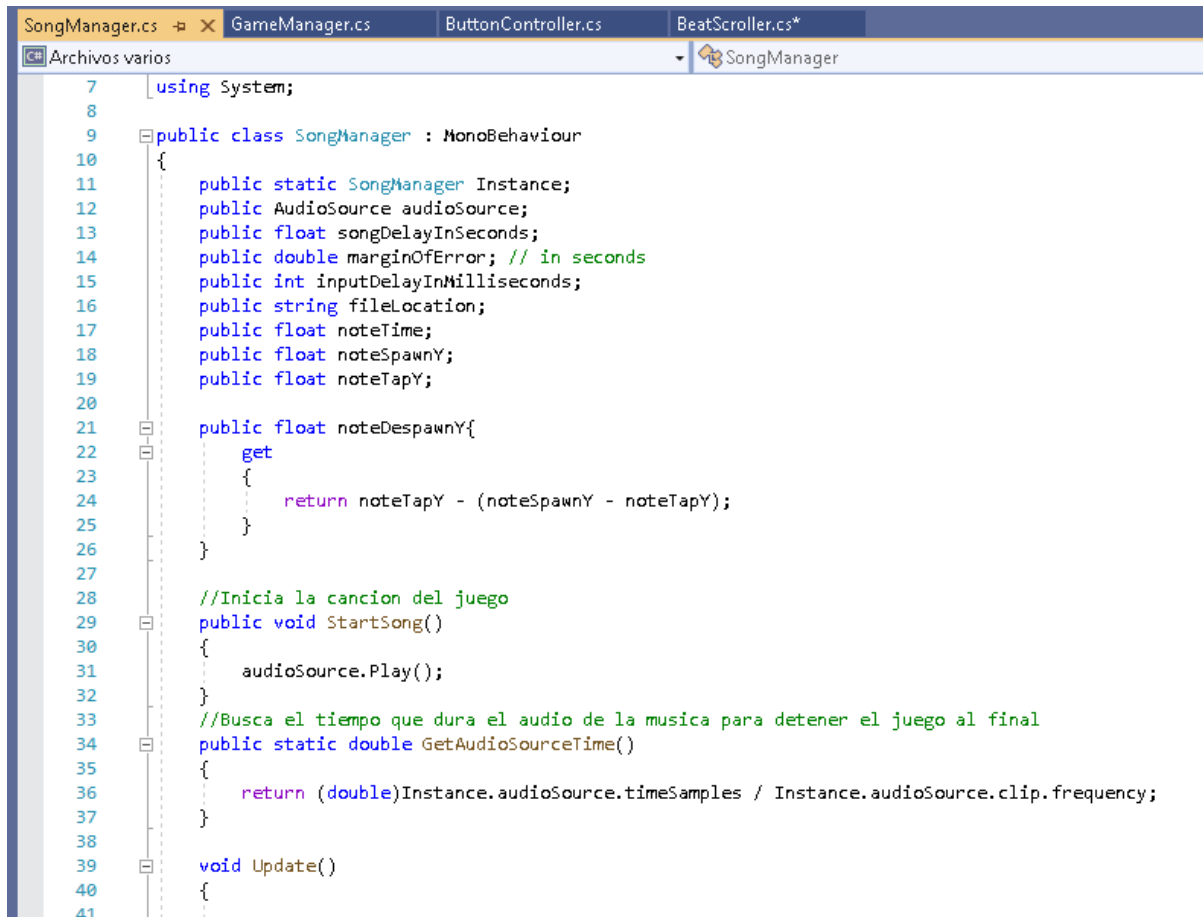
67 }
68 else
69 {
70     //muestra el resultado al finalizar la musica
71     // Y la calificacion Desde A hasta F
72     if ((!theMusic.isPlaying || !(theMusic.time != 0)) && (!resultsScreen.activeInHierarchy))
73     {
74         resultsScreen.SetActive(true);
75         normalsText.text = "" + NormalHits;
76         goodsText.text = "" + GoodHits;
77         perfectsText.text = "" + PerfectHits;
78         missesText.text = "" + MissedHits;
79
80         float totalHits = NormalHits + GoodHits + PerfectHits;
81         float percentHit = (totalHits / totalNotes) * 100f;
82
83         percentHitText.text = percentHit.ToString("F1") + "%";
84
85         string rankVal = "F";
86
87         if (percentHit >= 30)
88         {
89             rankVal = "D";
90             if (currentScore >= 550000)
91             {
92                 rankVal = "C";
93                 if (currentScore >= 750000)
94                 {
95                     rankVal = "B";
96                     if (currentScore >= 900000)
97                     {
98                         rankVal = "A";
99                         if (currentScore >= 950000)
100                         {
101                             rankVal = "S";
102                         }
103                     }
104                 }
105             }
106         }
107     }
108 }
```

```
GameManager.cs  ButtonController.cs  BeatScroller.cs*
Archivos varios  GameManager
100 {
101     rankVal = "S";
102 }
103 }
104 }
105 }
106 }
107 rankText.text = rankVal;
108 finalScoreText.text = currentScore.ToString();
109 }
110 }
111 }
112 //Esto calcula los puntos de al precionar las notas
113 //Desde aqui hasta el final segun la precision del jugador
114 public void NoteHit()
115 {
116     beatSfx.Play();
117     multiplierTracker++;
118     if ((currentMultiplier - 1) < multiplierThresholds.Length)
119     {
120         multiplierTracker++;
121         if (multiplierThresholds[currentMultiplier - 1] <= multiplierTracker)
122         {
123             multiplierTracker = 0;
124             currentMultiplier++;
125         }
126     }
127     scoreText.text = "" + currentScore;
128     multiText.text = "Score x" + currentMultiplier;
129 }
130 }
131 }
132 }
133 }
134 public void NormalHit()
135 {
```

```
GameManager.cs  ButtonController.cs  BeatScroller.cs*
Archivos varios  GameManager

133
134 public void NormalHit()
135 {
136     UnityEngine.Debug.Log("Normal Hit");
137     currentScore += scorePerNote * currentMultiplier;
138     NormalHits++;
139     NoteHit();
140 }
141
142 public void GoodHit()
143 {
144     UnityEngine.Debug.Log("Good Hit");
145     currentScore += scorePerGoodNote * currentMultiplier;
146     GoodHits++;
147     NoteHit();
148 }
149
150 public void PerfectHit()
151 {
152     UnityEngine.Debug.Log("Perfect Hit");
153     currentScore += scorePerPerfectNote * currentMultiplier;
154     PerfectHits++;
155     NoteHit();
156 }
157
158 public void NoteMissed()
159 {
160     UnityEngine.Debug.Log("missed");
161     MissedHits++;
162     currentMultiplier = 1;
163     multiplierTracker = 0;
164     multiText.text = "Score x" + currentMultiplier;
165 }
166
167
```

## Script para Controlar la musica:



```
7 using System;
8
9 public class SongManager : MonoBehaviour
10 {
11     public static SongManager Instance;
12     public AudioSource audioSource;
13     public float songDelayInSeconds;
14     public double marginOfError; // in seconds
15     public int inputDelayInMilliseconds;
16     public string fileLocation;
17     public float noteTime;
18     public float noteSpawnY;
19     public float noteTapY;
20
21     public float noteDespawnY{
22         get
23         {
24             return noteTapY - (noteSpawnY - noteTapY);
25         }
26     }
27
28     //Inicia la cancion del juego
29     public void StartSong()
30     {
31         audioSource.Play();
32     }
33     //Busca el tiempo que dura el audio de la musica para detener el juego al final
34     public static double GetAudioSourceTime()
35     {
36         return (double)Instance.audioSource.timeSamples / Instance.audioSource.clip.frequency;
37     }
38
39     void Update()
40     {
41     }
```

## 3.2 Prototipos

- Primer prototipo tocando cualquier botón para crear el mapa de piezas en el nivel
- Segundo prototipo con el primer nivel creado
- Tercer prototipo con Menues e instrucciones
- Cuarto prototipo Con el segundo nivel creado
- Quinto prototipo Prueba de Build para PC (Aprobado)
- Sexto prototipo Prueba para Build en Web (Aprobado con errores)
  - No carga los videos a menos que este en pantalla completa

## 3.3 Perfiles de Usuarios

- Personas mayores de 10 años de edad
- Apasionados por la música
- Personas que le guste poner a prueba sus sentidos

## 3.4 Usabilidad

- Utilizar Las teclas del teclado

- Tocar los botones de acuerdo a la melodía que viene

### **3.5 Test**

El público elegido para hacer la prueba del juego comprende entre 10 y 25 años y les gustó, pero recomendaron adaptarlo a música tradicional dominicana, como el merengue y la bachata.

### **3.6 Versiones de la Aplicación**

- Versión 0.6 : Música Electrónica y Rock , ideal para el juego, con 4 botones.